Package ‘condvis2’

June 28, 2019

**Title**  Conditional Visualization for Statistical Models

**Version**  0.1.0

**Description**  Constructs a shiny app function with interactive displays for conditional visualization of models, data and density functions. An extended version of package 'condvis'.


**License**  GPL (>= 2.0)

**Encoding**  UTF-8

**LazyData**  true

**Imports**  shiny, RColorBrewer, ggplot2, scales, cluster,

DendSer, methods, pplyr, kmed

**RoxygenNote**  6.1.1

**Suggests**  knitr, rmarkdown, hdrcde, scagnostics, keras,

kerlab, mclust, MASS, ks, mgcv, randomForest

**VignetteBuilder** knitr

**BugReports**  https://github.com/cbhurley/condvis2/issues

**URL**  https://github.com/cbhurley/condvis2

**NeedsCompilation**  no

**Author**  Catherine Hurley [aut, cre],

Mark O'Connell [aut],

Katarina Domijan [aut]

**Maintainer**  Catherine Hurley <catherine.hurley@mu.ie>

**Repository**  CRAN

**Date/Publication**  2019-06-28 15:40:10 UTC

**R topics documented:**

arrangeC ........................................... 2
conditionPlot ..................................... 3
condtour .......................................... 4
Make a list of variable pairings for condition selecting plots

Description

This function arranges a number of variables in pairs, ordered by their bivariate relationships. The goal is to discover which variable pairings are most helpful in avoiding extrapolations when exploring the data space. Variable pairs with strong bivariate dependencies (not necessarily linear) are chosen first. The bivariate dependency is measured using `savingby2d`. Each variable appears in the output only once.

Usage

```r
arrangeC(data, method = "default")
```

Arguments

- `data`: A dataframe
- `method`: The character name for the method to use for measuring bivariate dependency, passed to `savingby2d`.

Details

If `data` is so big as to make `arrangeC` very slow, a random sample of rows is used instead. The bivariate dependency measures are rough, and the ordering algorithm is a simple greedy one, so it is not worth allowing it too much time.

Value

A list containing character vectors giving variable pairings.

References

**conditionPlot**

*Plots a conditionPlot.*

---

**Description**

Plots a conditionPlot, showing one, two or many predictors. The predictor setting in varVal is drawn in magenta.

**Usage**

```r
conditionPlot(CVdata, var, varVal, pointColor = "steelblue", sim = NULL, resetpar = TRUE, plotrows = NULL)
```

**Arguments**

- `CVdata`: the dataset used for the fit
- `var`: one more condition vars. Draws a parallel coordinate plot for more than two.
- `varVal`: the current setting of the conditionvars, shown in magenta.
- `pointColor`: a color, vector of colors, or the name of variable to be used for coloring
- `sim`: If non-NULL should be a vector of similarity weights.
- `resetpar`: For use within shiny app.
- `plotrows`: If non-NULL should be a vector of case indices

**Examples**

```r
conditionPlot(mtcars, c("wt","hp"), c("wt"=3, "hp"=200), pointColor="am")
conditionPlot(mtcars, c("wt","hp"), mtcars[1,], pointColor="am")

#Calculate similarity using wt, hp observations from first case
sim <- similarityweight(mtcars[1, c("wt","hp")], mtcars[, c("wt","hp")], threshold=1)

# Marks points with black border with positive sim values. These are points within 1 (threshold) sd of pink cross.
conditionPlot(mtcars, c("wt","hp"), mtcars[1,], pointColor="am", sim=sim)

sim <- similarityweight(mtcars[1,], mtcars, threshold=2)
conditionPlot(mtcars, names(mtcars), mtcars[1,], sim=sim)
```
condtour

Tours of data space

Description

Tours of data space

Usage

randomPath(data, fits = NULL, length = 10, reorder = TRUE, conditionvars = NULL, ...)

kmeansPath(data, fits = NULL, length = 10, reorder = TRUE, conditionvars = NULL, ...)

pamPath(data, fits = NULL, length = 10, reorder = TRUE, conditionvars = NULL, ...)

claraPath(data, fits = NULL, length = 10, reorder = TRUE, conditionvars = NULL, ...)

fastkmedPath(data, fits = NULL, length = 10, reorder = TRUE, conditionvars = NULL, ...)

lofPath(data, fits, length = 10, reorder = TRUE, conditionvars = NULL, predictArgs = NULL)

diffsPath(data, fits, length = 10, reorder = TRUE, conditionvars = NULL, predictArgs = NULL)

createPath(data, score, length = 10, reorder = TRUE, conditionvars = NULL)

Arguments

data
fits
length
reorder
conditionvars
... predictArgs
score

A dataset
A model fit or list of fits
The length of path returns
If TRUE, points on the path are re-ordered so nearby points are close in the path
A vector of variable names. Some tours will limit calculation to this subset of variables.
other arguments, ignored
a list with one entry per fit, giving arguments for CVpredict
A vector of length equal to the nrows of data.
Details

kmeansPath works for both numeric and factors, which are converted to columns of indicators. pamPath is not recommended for large datasets, use claraPath instead. fits are used only in lofPath and diffitsPath. Paths are reordered using dser from package DendSer.

Value

A dataframe, which is the path

Functions

- randomPath: Returns a random path
- kmeansPath: Returns a path using kmeans centroids
- pamPath: Returns a path using pam medoids from package cluster
- claraPath: Returns a path using clara medoids from package cluster
- fastkmedPath: Returns a path using fastkmed from package kmmed
- lofPath: Returns a path showing biggest absolute residuals from fits.
- diffitsPath: Returns a path showing biggest difference in fits
- createPath: Returns a path showing highest scores

condvis

Creates Condvis Shiny app

Description

Creates Condvis Shiny app

Usage

condvis(data, model = NULL, response = NULL, sectionvars = NULL, conditionvars = NULL, predsInit = NULL, pointColor = c("steelblue", "grey0"), cPlotPCP = FALSE, cPlotn = 1000, orderConditionVars = arrangeC, threshold = 1, thresholdmax = 8 * threshold, linecols = NULL, showsim = NULL, theta3d = 45, phi3d = 20, dataplot = "pcp", tours = NULL, predictArgs = NULL, xlim = NULL, ylim = NULL, zlim = NULL, density = FALSE, showdata = density == FALSE, displayHeight = 950)

Arguments

data the dataset used for the fit. Should not have NAs for response, sectionvars or conditionvars.
model A fitted model or list of models. May be NULL.
response name of response variable. If null, tries to extract from model.
sectionvars  
names of sectionvars. If null, extracts from data.

conditionvars  
names of condition vars. If null, extracts from data.

predsInit  
Optionally provide starting value for some predictors. Defaults to median, or
mode for factors

pointColor  
a color, or the name of variable to be used for coloring. If the named variable is
numeric, it is first converted to a factor with 3 levels.

cPlotPCP  
if TRUE, conditionplots are drawn as a single PCP (for more than two conditionvars)

cPlotn  
Defaults to 1000. Shows a sample of this number of points in conditionplots. Non-numeric values are ignored.

orderConditionVars
If supplied, a function to order the Condition Vars

threshold  
used for similarity weights, defaults to 1.

thresholdmax  
maximum value allowed of threshold.

linecols  
vector of colors to be used for fits

showsIm  
if TRUE, shows sim in conditionplots with points/lines. Defaults to TRUE with
150 or fewer cases.

theta3d, phi3d  
Angles defining the viewing direction for 3d surface. theta3d gives the az-
imuthal direction and phi3d the colatitude. See persp.

dataplot  
"pcp" or "pairs". If CVfit is NULL, used to plot the data

tours  
A list of pre-calculated tours

predictArgs  
a list with one entry per fit, giving arguments for CVpredict

xlim  
passed on to sectionplot

ylim  
passed on to sectionplot

zlim  
passed on to sectionplot

density  
default FALSE. Use TRUE if model is a density function.

showdata  
defaults to density==TRUE. If FALSE, data on section not shown.

displayHeight  
supply a value for the display height

Examples

fit <- lm(mpg ~ wt+hp+am, data=mtcars)
if(interactive()){
  condvis(mtcars,fit, response="mpg", sectionvars="wt", conditionvars=c("am", "hp"), pointColor ="red")
}

createCVServer

Title Creates a shiny server

Description
Title Creates a shiny server

Usage
createCVServer(CVfit, CVdata = NULL, response = NULL, sectionvars, conditionvars, predsInit = NULL, cPlotPCP = FALSE, cPlotn = 1000, orderConditionVars, threshold = 1, thresholdmax, linecols = NULL, showsim = FALSE, dataplot = "pcp", probs, view3d, theta3d, phi3d, predictArgs, xlim = NULL, ylim = NULL, zlim = NULL, density = FALSE, showdata = TRUE)

Arguments

CVfit a list of fits
CVdata the dataset used for the fit
response name of response variable
sectionvars names of at most two sectionvars
conditionvars names of conditionvars
predsInit starting value for predicts. Defaults to median, or mode for factors
cPlotPCP if TRUE, conditionplots are drawn as a single PCP (for more than two conditionvars)
cPlotn Shows a sample of this number of points in conditionplots.
orderConditionVars

If supplied, a function to order the Condition Vars

threshold used for similarity weights, defaults to 1.
thresholdmax maximum value allowed of threshold.
linecols vector of colors to be used for fits
showsim if TRUE, shows sim in conditionplots with points
dataplot "pcp" or "pairs". If CVfit is NULL, used to plot the data
probs Logical; if TRUE, shows predicted class probabilities instead of just predicted classes.
view3d Logical; if TRUE, includes option for a three-dimensional regression surface if possible.
theta3d, phi3d Angles defining the viewing direction. theta3d gives the azimuthal direction and phi3d the colatitude. See persp.
predictArgs a list with one entry per fit, giving arguments for CVpredict
createCVUI

**Description**

Constructs UI for Condvis

**Usage**

```r
createCVUI(CVfit, data, sectionvars, preds = NULL, pointColor, 
           threshold = 1, thresholdmax, tours, probs, view3d)
```

**Arguments**

- `CVfit` a list of fits
- `data` a dataset
- `sectionvars` names of sectionvars
- `preds` names of predictors
- `pointColor` a color, or the name of variable to be used for coloring
- `threshold` used for similarity weights, defaults to 1.
- `thresholdmax` maximum value allowed of threshold.
- `tours` A list of pre-calculated tours
- `probs` Logical; if TRUE, shows predicted class probabilities instead of just predicted classes.
- `view3d` Logical; if TRUE, includes option for a three-dimensional regression surface if possible.

**Value**

a dataframe of conditions
CVpredict

A predict generic function for condvis

Description

A predict generic function for condvis

Usage

CVpredict(fit, newdata, ..., ptype = "pred", pthreshold = NULL,
ylevels = NULL, ptrans = NULL)

## Default S3 method:
CVpredict(fit, newdata, ..., ptype = "pred",
pthreshold = NULL, ylevels = NULL, ptrans = NULL)

## S3 method for class 'glm'
CVpredict(fit, ..., type = "response", ptype = "pred",
pthreshold = NULL, ylevels = NULL, ptrans = NULL)

## S3 method for class 'lda'
CVpredict(fit, ..., ptype = "pred", pthreshold = NULL,
ylevels = NULL, ptrans = NULL)

## S3 method for class 'qda'
CVpredict(fit, ..., ptype = "pred", pthreshold = NULL,
ylevels = NULL, ptrans = NULL)

## S3 method for class 'nnet'
CVpredict(fit, ..., type = NULL, ptype = "pred",
pthreshold = NULL, ylevels = NULL, ptrans = NULL)

## S3 method for class 'randomForest'
CVpredict(fit, ..., type = NULL, ptype = "pred",
pthreshold = NULL, ylevels = NULL, ptrans = NULL)

## S3 method for class 'rpart'
CVpredict(fit, ..., type = NULL, ptype = "pred",
pthreshold = NULL, ylevels = NULL, ptrans = NULL)

## S3 method for class 'tree'
CVpredict(fit, ..., type = NULL, ptype = "pred",
pthreshold = NULL, ylevels = NULL, ptrans = NULL)

## S3 method for class 'C5.0'
CVpredict(fit, ..., type = NULL, ptype = "pred",
pthreshold = NULL, ylevels = NULL, ptrans = NULL)
## S3 method for class 'svm'
CVpredict(fit, ..., type = NULL, ptype = "pred",
          pthreshold = NULL, ylevels = NULL, ptrans = NULL)

## S3 method for class 'gbm'
CVpredict(fit, ..., type = NULL, ptype = "pred",
          pthreshold = NULL, ylevels = NULL, n.trees = fit$n.trees,
          ptrans = NULL)

## S3 method for class 'loess'
CVpredict(fit, newdata = NULL, ...)

## S3 method for class 'ksvm'
CVpredict(fit, newdata, ..., type = NULL,
          ptype = "pred", pthreshold = NULL, ylevels = NULL, ptrans = NULL)

## S3 method for class 'glmnet'
CVpredict(fit, newdata, ..., type = "response",
          ptype = "pred", pthreshold = NULL, ylevels = NULL, ptrans = NULL,
          s = NULL, makex = NULL)

## S3 method for class 'cv.glmnet'
CVpredict(fit, newdata, ..., type = "response",
          ptype = "pred", pthreshold = NULL, ylevels = NULL, ptrans = NULL,
          makex = NULL)

## S3 method for class 'glmnet.formula'
CVpredict(fit, newdata, ..., type = "response",
          ptype = "pred", pthreshold = NULL, ylevels = NULL, ptrans = NULL,
          s = NULL)

## S3 method for class 'cv.glmnet.formula'
CVpredict(fit, newdata, ...,
          type = "response", ptype = "pred", pthreshold = NULL,
          ylevels = NULL, ptrans = NULL)

## S3 method for class 'keras.engine.training.Model'
CVpredict(fit, newdata, ...,
          ptype = "pred", pthreshold = NULL, ylevels = NULL, ptrans = NULL,
          batch_size = 32, response = NULL, predictors = NULL)

## S3 method for class 'kde'
CVpredict(fit, newdata = fit$x, ..., scale = TRUE)

## S3 method for class 'densityMclust'
CVpredict(fit, newdata = NULL, ...,
          ptype = "pred", pthreshold = NULL, ylevels = NULL, ptrans = NULL,
scale = TRUE)

## S3 method for class 'MclustDA'
CVpredict(fit, newdata, ..., ptype = "pred",
          pthreshold = NULL, ylevels = NULL, ptrans = NULL)

## S3 method for class 'MclustDR'
CVpredict(fit, newdata, ..., ptype = "pred",
          pthreshold = NULL, ylevels = NULL, ptrans = NULL)

## S3 method for class 'Mclust'
CVpredict(fit, newdata, ..., ptype = "pred",
          pthreshold = NULL, ylevels = NULL, ptrans = NULL)

## S3 method for class 'train'
CVpredict(fit, newdata, ..., type = "response",
          ptype = "pred", pthreshold = NULL, ylevels = NULL, ptrans = NULL)

## S3 method for class 'bartMachine'
CVpredict(fit, newdata, ..., type = NULL,
          ptype = "pred", pthreshold = NULL, ylevels = NULL, ptrans = NULL)

### Arguments

- **fit**: A fitted model
- **newdata**: Where to calculate predictions.
- **...**: extra arguments to predict
- **ptype**: One of "pred", "prob" or "probmatrix"
- **pthreshold**: Used for calculating classes from probs, in the two class case
- **ylevels**: The levels of the response, when it is a factor
- **ptrans**: A function to apply to the result
- **type**: For some predict methods
- **n.trees**: Used by CVpredict.gbm, passed to predict
- **s**: Used by CVpredict.glmnet and CVpredict.cv.glmnet, passed to predict
- **makex**: Used by CVpredict.glmnet and CVpredict.cv.glmnet. A function to construct xmatrix for predict.
- **batch_size**: Used by CVpredict.keras.engine.training.Model, passed to predict
- **response**: Used by CVpredict.keras.engine.training.Model. Name of response (optional)
- **predictors**: Used by CVpredict.keras.engine.training.Model. Name of predictors
- **scale**: Used by CVpredict for densities. If TRUE (default) rescales the conditional density to integrate to 1.
Details

This is a wrapper for predict used by condvis. When the model response is numeric, the result is a vector of predictions. When the model response is a factor the result depends on the value of ptype. If ptype=“pred”, the result is a factor. If also threshold is numeric, it is used to threshold a numeric prediction to construct the factor when the factor has two levels. For ptype=“prob”, the result is a vector of probabilities for the last factor level. For ptype=“probmatri”, the result is a matrix of probabilities for each factor level.

Value

a vector of predictions, or a matrix when type is "probmatri"

Methods (by class)

• default: CVpredict method
• glm: CVpredict method
• lda: CVpredict method
• qda: CVpredict method
• nnet: CVpredict method
• randomForest: CVpredict method
• rpart: CVpredict method
• tree: CVpredict method
• C5.0: CVpredict method
• svm: CVpredict method
• gbm: CVpredict method
• loess: CVpredict method
• ksvm: CVpredict method
• glmnet: CVpredict method
• cv.glmnet: CVpredict method
• glmnet.formula: CVpredict method
• cv.glmnet.formula: CVpredict method
• keras.engine.training.Model: CVpredict method
• kde: CVpredict method
• densityMclust: CVpredict method
• MclustDA: CVpredict method
• MclustDR: CVpredict method
• Mclust: CVpredict method
• train: CVpredict method
• bartMachine: CVpredict method
Examples

# Fit a model.
f <- lm(Fertility ~ ., data=swiss)
CVpredict(f)

# Fit a model with a factor response
swiss1 <- swiss
swiss1$Fertility <- cut(swiss$Fertility, c(0,80,100))
levels(swiss1$Fertility) <- c("lo", "hi")
f <- glm(Fertility ~ ., data=swiss1, family="binomial")
CVpredict(f) # by default gives a factor
CVpredict(f, ptype="prob") # gives prob of level hi
CVpredict(f, ptype="probmatrix") # gives prob of both levels

Description

Interpolation

Usage

pathInterpolate(x, ninterp = 4)

## Default S3 method:
pathInterpolate(x, ninterp = 4L)

## S3 method for class 'factor'
pathInterpolate(x, ninterp = 4L)

## S3 method for class 'data.frame'
pathInterpolate(x, ninterp = 4L)

Arguments

x a numeric or factor vector or dataframe
ninterp number of interpolated steps

Value

interpolated version of x

Methods (by class)

- default: Default interpolate method
- factor: pathInterpolate method for factor
- data.frame: pathInterpolate method for data.frame
plotTourDiagnostics

*Plots diagnostics for the tour supplied*

**Description**

Plots diagnostics for the tour supplied

**Usage**

```r
plotTourDiagnostics(path, data, pathlen = nrow(path), threshold = 1,
                    which = 1:3, ...)  
```

**Arguments**

- `path`: the tour
- `data`: the dataset
- `pathlen`: the pathlength
- `threshold`: used for similarityweight
- `which`: subset of 1:3
- `...`: other args for similarityweight

**Details**

The first plot shows approximately how much data are visible on each section, the second shows what proportion of data are visited by the tour, and the third a density estimate of max similarity values.

**Value**

Table of max sims attained.

---

sectionPlot

*Plots the main condvis display*

**Description**

The section plot relates a fit or fits to one or two predictors (`sectionvar`), for fixed values of other predictors in `conditionvals`. 
Usage

sectionPlot(CVdata, CVfit = NULL, response = NULL, preds, sectionvar, conditionvals, pointColor = "steelblue", sim = NULL, threshold = 1, linecols = NULL, dataplot = "pcp", gridsize = 50, probs = FALSE, view3d = FALSE, theta3d = 45, phi3d = 20, xlim = NULL, ylim = NULL, zlim3d = 45, phi3d = 20, x3lim = NULL, xlim = NULL, zlim = NULL, pointSize = 2, predictArgs = NULL, resetpar = TRUE, density = FALSE, showdata = density == FALSE, returnCoords = FALSE)

Arguments

CVdata the dataset used for the fit
CVfit a fit or list of fits
response name of response variable
preds names of predictors
sectionvar section variable
conditionvals conditioning values. A vector/list or dataframe with one row
pointColor a color, vector of colors, or the name of variable to be used for coloring
sim vector of similarity weights
threshold used for similarity weights, defaults to 1.
linecols vector of line colours
dataplot "pcp" or "pairs". If CVfit is NULL, used to plot the data
gridsize used to construct grid of fitted values.
probs Logical; if TRUE, shows predicted class probabilities instead of just predicted classes. Only available with two numeric sectionvars and the model's predict method provides this.
view3d Logical; if TRUE plots a three-dimensional regression surface if possible.
theta3d, phi3d Angles defining the viewing direction. theta3d gives the azimuthal direction and phi3d the colatitude. See persp.
xlim passed on to plot
ylim passed on to plot
zlim passed on to plot
pointSize used for points
predictArgs a list with one entry per fit, giving arguments for predict
resetpar When TRUE (the default) resets pars after drawing.
density default FALSE. Use TRUE if model is a density function.
showdata If FALSE, data on section not shown.
returnCoords If TRUE, returns coordinates for some plots
The similarity weight is a measure of how similar a given observation is to a reference point in the data space. Observations that are very similar to the reference point are given a weight of 1, while observations that are dissimilar are given a weight of 0. This function is particularly useful for calculating weights for subsequent plots, such as parallel coordinate plots or pairs plots.

**Description**

Calculate the similarity weight for a set of observations, based on their distance from some arbitrary points in data space. Observations which are very similar to the point under consideration are given weight 1, while observations which are dissimilar to the point are given weight zero.

**Usage**

```r
similarityweight(x, data, threshold = NULL, distance = "euclidean",
lambda = NULL, scale = TRUE)
```
similarityweight

Arguments

- **x**: A dataframe describing arbitrary points in the space of the data (i.e., with same colnames as data).
- **data**: A dataframe representing observed data.
- **threshold**: Threshold distance outside which observations will be assigned similarity weight zero. This is numeric and should be > 0. Defaults to 1.
- **distance**: The type of distance measure to be used, currently just three types of Minkowski distance: "euclidean" (default), "maxnorm" and "manhattan".
- **lambda**: A constant to multiply by the number of categorical mismatches, before adding to the Minkowski distance, to give a general dissimilarity measure. If left NULL, behaves as though lambda is set larger than threshold, meaning that one factor mismatch guarantees zero weight.
- **scale**: defaults to TRUE, in which case numeric variables are scaled to unit sd.

Details

Similarity weight is assigned to observations based on their distance from a given point. The distance is calculated as Minkowski distance between the numeric elements for the observations whose categorical elements match, with the option to use a more general dissimilarity measure comprising Minkowski distance and a mismatch count.

Value

A numeric vector or matrix, with values from 0 to 1. The similarity weights for the observations in data arranged in rows for each row in x.

References


Examples

```r
## Say we want to find observations similar to the first observation.
## The first observation is identical to itself, so it gets weight 1. The
## second observation is similar, so it gets some weight. The rest are more
## different, and so get zero weight.

data(mtcars)
similarityweight(x = mtcars[1, ], data = mtcars)

## By increasing the threshold, we can find observations which are more
## approximately similar to the first row. Note that the second observation
## now has weight 1, so we lose some ability to discern how similar
## observations are by increasing the threshold.

similarityweight(x = mtcars[1, ], data = mtcars, threshold = 5)
```
```r
similarityweight(mtcars[1:2, ], mtcars, threshold = 3)
```

---

**weightcolor**  
*Fade colours according to a weight vector*

**Description**

The colours whose weights are less than 1 are diluted. Colours whose weight is zero are returned as white, other weights are grouped in `nlevels` groups and colours diluted proportionally.

**Usage**

```r
weightcolor(col, weights, nlevels = 5)
```

**Arguments**

- `col`: A vector of colour
- `weights`: A vector of weights, values between 0 and 1
- `nlevels`: The number of groups

**Value**

A vector of colours
Index

arrangeC, 2
claraPath (condtour), 4
conditionPlot, 3
condtour, 4
condvis, 5
createCVServer, 7
createCVUI, 8
createPath (condtour), 4
CVpredict, 9
diffsPath (condtour), 4
fastkmedPath (condtour), 4
kmeansPath (condtour), 4
lofPath (condtour), 4
pamPath (condtour), 4
pathInterpolate, 13
persp, 6, 7, 15
plotTourDiagnostics, 14
randomPath (condtour), 4
sectionPlot, 14
similarityweight, 16
weightcolor, 18